NAME

perl5131delta - what is new for perl v5.13.1

DESCRIPTION

This document describes differences between the 5.13.0 release and the 5.13.1 release.

If you are upgrading from an earlier release such as 5.10, first read *perl5120delta*, which describes differences between 5.10 and 5.12.

Incompatible Changes

"\cX"

The backslash-c construct was designed as a way of specifying non-printable characters, but there were no restrictions (on ASCII platforms) on what the character following the c could be. Now, that character must be one of the ASCII characters.

localised tied hashes, arrays and scalars are no longed tied

In the following:

```
    tie @a, ...;
    {
 local @a;
 # here, @a is a now a new, untied array
    }
    # here, @a refers again to the old, tied array
```

The new local array used to be made tied too, which was fairly pointless, and has now been fixed. This fix could however potentially cause a change in behaviour of some code.

given return values

Starting from this release, `given` blocks returns the last evaluated expression, or an empty list if the block was exited by `break`. Thus you can now write:

```
    my $type = do {
     given ($num) {
      break     when undef;
      'integer' when /^[+-]?[0-9]+$/;
      'float'   when /^[+-]?[0-9]+(?:\.[0-9]+)?$/;
      'unknown';
     }
    };
```

See *"Return value" in perlsyn* for details.

Core Enhancements

Exception Handling Reliability

Several changes have been made to the way `die`, `warn`, and $@ behave, in order to make them more reliable and consistent.

When an exception is thrown inside an `eval`, the exception is no longer at risk of being clobbered by code running during unwinding (e.g., destructors). Previously, the exception was written into $@ early in the throwing process, and would be overwritten if `eval` was used internally in the destructor for an object that had to be freed while exiting from the outer `eval`. Now the exception is written into $@ last thing before exiting the outer `eval`, so the code running immediately thereafter can rely on the value in $@ correctly corresponding to that `eval`.

Likewise, a `local $@` inside an `eval` will no longer clobber any exception thrown in its scope.

Previously, the restoration of $@ upon unwinding would overwrite any exception being thrown. Now the exception gets to the `eval` anyway. So `local $@` is safe inside an `eval`, albeit of rather limited use.

Exceptions thrown from object destructors no longer modify the $@ of the surrounding context. (If the surrounding context was exception unwinding, this used to be another way to clobber the exception being thrown. Due to the above change it no longer has that significance, but there are other situations where $@ is significant.) Previously such an exception was sometimes emitted as a warning, and then either string-appended to the surrounding $@ or completely replaced the surrounding $@, depending on whether that exception and the surrounding $@ were strings or objects. Now, an exception in this situation is always emitted as a warning, leaving the surrounding $@ untouched. In addition to object destructors, this also affects any function call performed by XS code using the `G_KEEPERR` flag.

$@ is also no longer used as an internal temporary variable when preparing to `die`. Previously it was internally necessary to put any exception object (any non-string exception) into $@ first, before it could be used as an exception. (The C API still offers the old option, so an XS module might still clobber $@ in the old way.) This change together with the foregoing means that, in various places, $@ may be observed to contain its previously-assigned value, rather than having been overwritten by recent exception-related activity.

Warnings for `warn` can now be objects, in the same way as exceptions for `die`. If an object-based warning gets the default handling, of writing to standard error, it will of course still be stringified along the way. But a `$SIG{__WARN__}` handler will now receive an object-based warning as an object, where previously it was passed the result of stringifying the object.

Modules and Pragmata

Updated Modules

`Errno`

The implementation of `Errno` has been refactored to use about 55% less memory. There should be no user-visible changes.

Perl 4 `.pl` libraries

These historical libraries have been minimally modified to avoid using `$[`. This is to prepare them for the deprecation of `$[`.

`B::Deparse`

A bug has been fixed when deparsing a nextstate op that has both a change of package (relative to the previous nextstate), or a change of `%^H` or other state, and a label. Previously the label was emitted first, leading to syntactically invalid output because a label is not permitted immediately before a package declaration, **BEGIN** block, or some other things. Now the label is emitted last.

Removed Modules and Pragmata

The following modules have been removed from the core distribution, and if needed should be installed from CPAN instead.

`Class::ISA`

`Pod::Plainer`

`Switch`

The removal of `Shell` has been deferred until after 5.14, as the implementation of `Shell` shipped with 5.12.0 did not correctly issue the warning that it was to be removed from core.

New Documentation

perlgpl

perlgpl has been updated to contain GPL version 1, as is included in the *README* distributed with perl.

Selected Bug Fixes

- Naming a deprecated character in \N{...} will not leak memory.
- FETCH is no longer called needlessly on some tied variables.
- The trie runtime code should no longer allocate massive amounts of memory, fixing #74484.

Changed Internals

- The protocol for unwinding the C stack at the last stage of a `die` has changed how it identifies the target stack frame. This now uses a separate variable `PL_restartjmpenv`, where previously it relied on the `blk_eval.cur_top_env` pointer in the `eval` context frame that has nominally just been discarded. This change means that code running during various stages of Perl-level unwinding no longer needs to take care to avoid destroying the ghost frame.
- The format of entries on the scope stack has been changed, resulting in a reduction of memory usage of about 10%. In particular, the memory used by the scope stack to record each active lexical variable has been halved.
- Memory allocation for pointer tables has been changed. Previously `Perl_ptr_table_store` allocated memory from the same arena system as SV bodies and HEs, with freed memory remaining bound to those arenas until interpreter exit. Now it allocates memory from arenas private to the specific pointer table, and that memory is returned to the system when `Perl_ptr_table_free` is called. Additionally, allocation and release are both less CPU intensive.
- A new function, Perl_magic_methcall has been added that wraps the setup needed to call a magic method like FETCH (the existing S_magic_methcall function has been renamed S_magic_methcall1).

Deprecations

The following items are now deprecated.

`Perl_ptr_table_clear`

`Perl_ptr_table_clear` is no longer part of Perl's public API. Calling it now generates a deprecation warning, and it will be removed in a future release.

Acknowledgements

Perl 5.13.1 represents thirty days of development since Perl 5.13.0 and contains 15390 lines of changes across 289 files from 34 authors and committers.

Thank you to the following for contributing to this release:

Ã†var ArnfjÃ¶rÃ° Bjarmason, Arkturuz, Chris 'BinGOs' Williams, Craig A. Berry, Curtis Jewell, Dan Dascalescu, David Golden, David Mitchell, Father Chrysostomos, Gene Sullivan, gfx, Gisle Aas, H.Merijn Brand, James E Keenan, James Mastros, Jan Dubois, Jesse Vincent, Karl Williamson, Leon Brocard, Lubomir Rintel (GoodData), Nicholas Clark, Philippe Bruhat (BooK), Rafael Garcia-Suarez, Rainer Tammer, Ricardo Signes, Richard Soderberg, Robin Barker, Ruslan Zakirov, Steffen Mueller, Todd Rinaldo, Tony Cook, Vincent Pit, Zefram

Reporting Bugs

If you find what you think is a bug, you might check the articles recently posted to the comp.lang.perl.misc newsgroup and the perl bug database at http://rt.perl.org/perlbug/ . There may

also be information at http://www.perl.org/ , the Perl Home Page.

If you believe you have an unreported bug, please run the **perlbug** program included with your release. Be sure to trim your bug down to a tiny but sufficient test case. Your bug report, along with the output of `perl -V`, will be sent off to perlbug@perl.org to be analysed by the Perl porting team.

If the bug you are reporting has security implications, which make it inappropriate to send to a publicly archived mailing list, then please send it to perl5-security-report@perl.org. This points to a closed subscription unarchived mailing list, which includes all the core committers, who be able to help assess the impact of issues, figure out a resolution, and help co-ordinate the release of patches to mitigate or fix the problem across all platforms on which Perl is supported. Please only use this address for security issues in the Perl core, not for modules independently distributed on CPAN.

SEE ALSO

The *Changes* file for an explanation of how to view exhaustive details on what changed.

The *INSTALL* file for how to build Perl.

The *README* file for general stuff.

The *Artistic* and *Copying* files for copyright information.